Deliverable 2

1. I am changing my project to a reinforcement learning project in which I will make a bipedal walker move forward as fast as possible. The first goal would be to reach the end of the environment by moving there without falling down, and the second goal would be to reach the end of the environment as fast as possible.
2. I am not using a dataset for this problem, so I will talk about the environment I will be using to complete this project. I will use OpenAI’s Gym toolkit in order to make the implementation of the environment easy. More specifically, this environment uses Box2D’s physics engine to simulate the environment. It has 24 different observations in the observation space and has 4 different actions, so it is decently complex but not too much either. The agent is rewarded for moving forward, punished for falling, and marginally penalized for applying motor torque, meaning in theory the agent will minimize movement as it learns.
3. The machine learning model I will use is either a Q-learning algorithm, or a deep Q-learning algorithm. I have not reached the point yet where I will know which is better, but essentially if there are too many state action pairs to realistically handle in the q-learning table, then I will use TensorFlow or some other NN library to predict the Q value based on the state action pair input.
4. What’s nice about reinforcement learning is that it is very easy to visualize progress. The one graph that is probably useful would be the reward vs generation graph. At this point, I’ve implemented q-learning with a simpler environment: a car trying to get up a hill. The model worked very well there, requiring only a few hundred generations to reach the flag at the top of the hill. It also was able to find what looked like a near optimal solution after a few thousand generations. (I’ve uploaded the .ipynb file of my q-learning algorithm to my github (it probably won’t work on colab, but it’ll work on a local jupyter notebook)).
5. My next steps are pretty clear-cut: I will first implement q-learning with my desired environment and see if regular RL will work there. I will then tweak the hyperparameters until I get a decent solution. From there, I will probably still try to implement a deep q-learning algorithm to see how well it does.